# OPTIMAL SENSOR PLACEMENT

BY

XUANDONG XU

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2015

Urbana, Illinois

Adviser:

Assistant Professor Mohamed Ali Belabbas

## Abstract

In this thesis we explore the problem of finding optimal sensor/actuator locations to achieve the minimum square error/least effort. The solution for the optimal sensor/actuator is often combinational, this means in order to solve for the solution we have to look at many parameters in the system and those parameters change frequently. In this thesis, we propose two methods to achieve this goal: the first one is based on gradient flow differential in which it provides the global optimal solution for the placement, and the second one is based on the evaluation of the Hessian matrix at the critical points. The optimal sensor/actuator location found using the gradient flow or Hessian matrix is usually not sparse. However in practical settings, the optimal sensor/actuator locations are often determined by discrete numbers such as ones and zeroes. We then propose some methods of relating the optimal sensor/actuator locations found using the gradient flow or Hessian matrix to the optimal sensor locations in the practical settings. Next we test the performance of the method we proposed by comparing the square error/effort of the system using the optimal sensor/actuator locations we found and the optimal sensor locations in the practical settings in multiple dimensions and with selected number of sensors/actuators.

# Acknowledgment

There are so many people to thank who have made this thesis possible. First and foremost, many thanks to my thesis advisor Professor Mohammed Ali Belabbas. His guidance has been pivotal and always encouraging.

# Contents

1. Introduction ..... 1
1.1 Brief Overview ..... 1
1.2 Proposed Approach ..... 3
1.3 Thesis Outline ..... 3
2. Prior and Related Work ..... 4
3. Gradient Approach ..... 6
3.1 Background Information of the Model ..... 6
3.2 Algorithm Description ..... 7
3.3 Numerical Result ..... 9
4. Hessian Approach ..... 12
4.1 Background Information ..... 12
4.2 Algorithm Description ..... 12
4.3 Numerical Result ..... 14
5. Relationship between Optimal Model and Optimal Discrete Model ..... 16
5.1 Background Information ..... 16
5.2 Proposed Approaches ..... 16
6. Simulations and Result ..... 18
6.1 Experimental Setup ..... 18
6.2 Trace of Error Comparison (Number of Sensor Fixed) ..... 18
6.3 Trace of Error Comparison (Dimension Fixed) ..... 20
6.4 Runtime Comparison ..... 20
7. Conclusions and Future Work ..... 21
References ..... 22

# 1. Introduction

## 1.1 Brief Overview

**Optimal observation model.** The internal states of the system are often perturbed by various kinds of noise. Sometimes when we want to estimate these internal states we can only estimate them from partial and noisy environments coming from a set of sensors. The issue here is choosing an observation model that can minimize the error between the estimated signal and the real internal state signal.

There are already some well-developed methods in control theory to design the optimal estimator given the observation model. Generally these estimators are called observers of the system. Consider the general stochastic system with some disturbance:

$$\dot{X} = Ax + Bu + FW$$
$$Y = CX + V$$

where $X$ represent the states, $u$ is the input, $W$ represents disturbance that affects the dynamic of the system and $V$ represents measurements noise at the observer end. In this thesis we assume the disturbances $W$ and measurement noises $V$ are independent zero-mean Gaussian white noises with covariance $R_w$ and $R_v$. The general approach of finding the optimal estimation problem is to find the estimate $\hat{x}$ that minimizes the mean square error $E\{(X(t) - \hat{X}(t))(X(t) - \hat{X}(t))^{\intercal}\}$ given $\{Y(\tau): 0 \leq \tau \leq t\}$. This type of problem can be viewed as solving the *least squares* problem: given all previous data $Y(t)$, find an estimate $\hat{X}$ which satisfy the dynamics of the system as well minimizing the square error of internal states.

The optimal estimator has the form of a linear observer [1]

$$\dot{\hat{X}} = A\hat{X} + BU + L(Y - C\hat{X})$$

where $L = PC^T R_v^{-1}$ and $P = E\{(X(t) - \hat{X}(t))(X(t) - \hat{X}(t))^T\}$ satisfies the *Riccati differential equation* in steady state:

$$0 = AP + PA^T - PC^T R_v^{-1} CP + FR_w F^T$$

It is worth mentioning that the solution to *Ricatti equation* does not always exist. In fact, if the pair $(A, C)$ is not observable then one might not be able to design a convergent observer.

The problem with the Kalman-Bucy filter is that it gives you the optimal measurement in terms of mean square error but only for a fixed observation model. What we want to explore is finding the best observation model that minimizes the estimation error. In this thesis we choose the trace of the

estimation error as our measurement (bear in mind that there are other choices of measurements available)

$$tr(K) = \sum_i E\{(X(t) - \hat{X}(t))(X(t) - \hat{X}(t))^\intercal\}$$

It is also intuitive to see that if the norm of $C$ increases, the estimation error will decrease [2]. Given this condition, it is natural for us to find the optimal $C$ that minimizes the trace of the estimation error given the norm of $C$ is fixed.

After finding the optimal $C$ or the suboptimal $C$ (given some relaxation conditions), the next step would be relating this solution to a discrete observation model in a real-world scenario.

**Optimal actuator model.** There exists a duality between observability and controllability. We can observe some similarities between optimal actuator model and optimal observation model because of this duality. Consider the linear dynamic system

$$\dot{x} = Ax + Bu + v$$
$$y = Cx + w$$

The optimal linear quadratic controller minimize the following cost function

$$J = E\left(x^\intercal(T)Fx(T) + \int_0^T x^\intercal(t)Q(t)x(t) + u^\intercal(t)R(t)u(t)dt\right)$$
$$F \geq 0, \;\; Q(t) \geq 0, \;\; R(t) \geq 0$$

The final time $T$ can be either finite or infinite. In our case, we just need to know how the system behaves at steady state so we set $T$ to be infinite. The control law that minimizes the cost function takes the form $u = -L(t)x$, where $L(t) = R^{-1}(t)B^T(t)K(t)$. The $K(t)$ in the last equation also obeys the *Riccati differential equation*

$$0 = A^\intercal K + KA - KBR^{-1}B^\intercal K + Q$$

Let $K(t, t_1)$ denote the solution to the RDE with time horizon $t_1$, so the optimal control may be written as $u = -R^{-1}B^\intercal K(t, t_1)x(t)$. Suppose that $K(t, t_1) \to \overline{K}$ as $t_1 \to \infty$, where $\overline{K}$ is an matrix independent of time. Then the limiting control law $u(t) = -R^{-1}B^\intercal \overline{K}x(t)$ is optimal for the infinite horizon control problem. Letting $V^0$ denote the optimal cost for the infinite horizon problem, it follows that $x^\intercal K(0, t_1)x \leq V^0$ for all $x$ and $t_1$[3].

Thus given an initial condition $x_0$, the optimal expected cost of returning to zero with this control law is thus $J(x_0) = x_0^\intercal K x_0$. If the initial condition is distributed according to an arbitrary rotationally invariant space with density $g(r)dr$, where $r = \|x\|$, then the optimal expected cost is thus

$$\mathbb{E}J = \int_0^\infty g(r)drtr(K)$$

The design goal here is thus to find an actuator model $B$ such that the trace of $K$ is minimized. Note that the above actuator design only handles non-rotationally invariant distribution on the initial conditions [4].

## 1.2 Proposed Approach

In this thesis, we will discuss two approach of finding the optimal observation model. The first approach finds the solution via gradient differential. This solution is proven to be the optimal observation model $c_{opt}$ given the norm of observation model is fixed [4]. The second approach finds the solution via random testing and evaluating the eigenvalues of the Hessian matrix. This solution can be suboptimal because we only stop when all the eigenvalues are close to zero.

The next step would be finding the pseudo optimal discrete observation model $c_{pseudo}$ based on the previously found optimal observation model $c_{opt}$. We will also compare the result with the optimal discrete model $c_{discrete}$ and see how each strategy performs.

A visual representation of how the process will go is provided in Figure 1.

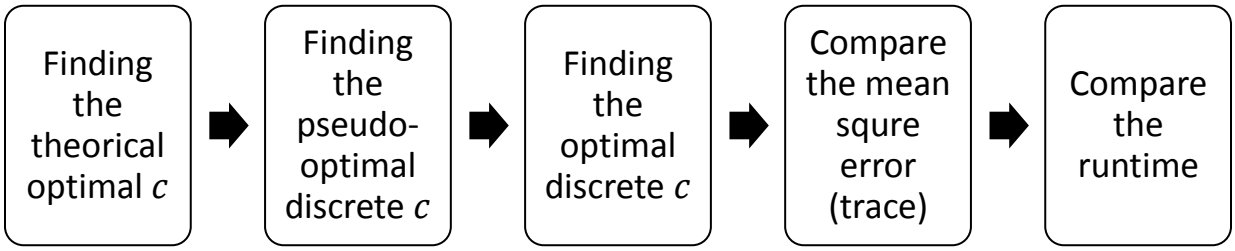


**Figure 1. Process overview.**

## 1.3 Thesis Outline

The remainder of this thesis is organized as follows. Chapter 2 serves as a motivation and discusses related work to this thesis. Chapter 3 discusses in detail how the gradient approach of finding the solution is implemented. Chapter 4 discusses how the Hessian approach of finding the suboptimal solution is implemented and talks a bit more of its limitations. Chapter 5 shows a couple of strategies of finding the pseudo-optimal discrete model $c_{pseudo}$ with the $c_{opt}$ previously found using either gradient differential or Hessian approach. Chapter 6 shows the performance results from simulations and compares the runtime. Chapter 7 concludes our result and offers directions of future work.

## 2. Prior and Related Work

The optimal sensor/actuator placement falls into the category of optimal control problems. One application of this work is in the field of active noise regulator and tracking. The noise of the sound is detected by microphones (sensors), and loudspeakers (actuators) to generate a secondary acoustic field that interacts destructively with the primary noise field [5]. To reduce the noise, many researchers have devised strategies using techniques in the frequency domain. However active noise reduction can also be achieved by employing state-space formulation of the optimal sensor/actuator locations. In [5], the authors presented two active control strategies: One is based on LQT [6-8] formulation which models the primary noise as a plane wave. This strategy is essentially suitable for actuator placement. The second strategy incorporates the primary sound field dynamics in the state equation. The advantage of the second strategy is that one can determine both the optimal sensor and actuator placement if the given system is not complicated to solve.

In another paper, Morris [9] considered the problem using approximations to determine the optimal actuator placement for linear quadratic optimal control. In practice the equation for the optimal control cannot be solved easily and the solution is done by approximation. The use of approximations usually leads to solving a finite-dimensional ARE for a finite-rank approximation. However, the author also mentioned that using approximation to determine the best actuator location can introduce an additional layer of optimization. The conclusion is that the optimal cost and location from the approximating sequence may not converge to the exact cost and location. In our thesis, we will show an example of this by demonstrating the Hessian method of finding optimal sensor location.

Another approach that was discussed in [10] is to use a genetic algorithm [11-14] to find out the suboptimal solution for the sensor placement. In most real-life problems, near optimal solutions that can be generated quickly are more desirable than optimal solutions which require a huge amount of time. The GA algorithm first forms a set of solutions randomly. Then the algorithm would evaluate the fitness of these solutions. Upon assessing the fitness of all the potential solutions, a new generation of individuals is created from the current set of solutions by using crossover and mutation operator. The algorithm is terminated using one of two stopping criteria: either the number of generations reaches a predefined maximum value $N_G$ or the current solution does not yield sufficient improvement $u_{min}$ compared with the performance reached by the previous generation of solutions. The approach discussed in [10] also selected the error covariance matrix as the performance variable. However, the

performance indicator is evaluated over the entire time-horizon instead at each instance of time. In fact, if the system is continuous and dynamic, then this kind of performance indicator is more appropriate.

Compared to the rest of the methods, the gradient approach in this thesis only finds the optimal sensor location that minimizes the trace of the error covariance at steady state given a fixed norm. It does not optimize the cost and the error at the same time, nor does it evaluate performance over the entire time horizon. Problems that might benefit from gradient approach analysis include optimal scheduling and design of linear systems [15], the joint optimal measurement and control design [16] or the estimation of complex systems [17].

# 3. Gradient Approach

## 3.1 Background Information of the Model

For simplicity, we only consider the linear time invariant system in our simulation. But this will also work for systems that are time variant. Since the gradient flow utilizes the *ode45* function in MATLAB, this means we will need an initial condition on the observation model and we will also need to iterate several times until we find a solution that converges. For the source of the disturbance in our model, we set it to be stationary and only dependent on the dimension of the system. In our case we assume as the dimension grows the covariance of the process noise will shrink. Intuitively this makes sense because as dimension grows, the time required to solve for the optimal solution will increase, as time increases we have more time to observe the model, and the longer we observer the model the accuracy of our solution will increase. We will also place a constant parameter called $\gamma$, or SNR (signal-to-noise ratio) in our observation model. The SNR parameter will act as the norm for the observation model. As we mentioned in our introduction, as the norm increases the estimation error will decrease. So it is likely that increasing the norm will increase the positive performance of the algorithm. However we cannot raise this parameter to be too big because the algorithm only guarantees a unique minimum when $\gamma$ is small.

The model we will be using is the linear Stratonovich [18] stochastic differential equation:

$$dx = Axdt + budt + Gdw$$
$$dy = cxdt + dv$$

where $dw$ and $dv$ are independent Wiener processes and $A \in \mathbb{R}^{n\times n}$, $b \in \mathbb{R}^{n\times q}$, $G \in \mathbb{R}^{n\times r}$ and $c \in \mathbb{R}^{p\times n}$. The goal is to design an estimator for $x$ given past observation $y([0,t])$. Since the Kalman filter is the optimal estimator in terms of mean square error, we will be using that model as well

$$d\hat{x} = A\hat{x}dt - Kc^{\intercal}R_v^{-1}(dy - c\hat{x}dt) + budt$$

where the matrix $K$ is the symmetric positive definite solution of the following *Riccati equation*:

$$KA^{\intercal} + AK - Kc^{\intercal}cK + GG^{\intercal} = 0$$

The right-hand side of the above equation is $0$ since we are only evaluating it at steady state. The solution $K$ is also the steady-state covariance matrix of the estimation error [4]

$$K = \mathbb{E}((x-\hat{x})(x-\hat{x})^{\intercal})$$

***Sketch of proof.*** Set the Kalman gain to be $L = Kc^{\intercal}R_v^{-1}$. The error dynamics are therefore:

$$\dot{E} = \dot{x} - \dot{\hat{x}} = (A - Lc)E + \zeta, \quad \zeta = Gw - Lv, \quad Cov(\zeta) = GR_wG^{\intercal} + LR_vL^{\intercal}$$

where $R_w$ and $R_v$ are the covariance of the disturbance and noise. The covariance matrix $K$ for this process satisfies

$$\begin{aligned}\dot{K} &= (A - Lc)K + K(A - Lc)^\intercal + GR_wG^\intercal + R_v \\ &= AK + KA^\intercal + GR_wG^\intercal - LcK - Kc^\intercal L^\intercal + LR_vL^\intercal \\ &= AK + KA^\intercal + GR_wG^\intercal + (LR_v - KC^\intercal)R_v^{-1}(LR_v + Kc^\intercal)^\intercal - Kc^\intercal R_v CP\end{aligned}$$

Since we want to find $L$ such that $K$ is as small as possible, we select $L$ such that $\dot{K}$ decreases by the maximum amount possible at each instant in time [19]. This is accomplished by setting

$$LR_v = KC^\intercal \Rightarrow L = Kc^\intercal R_v^{-1}$$

## 3.2 Algorithm Description

In this section, we will present a geometric method of finding the optimal observation model. The basic idea is to find another pseudo-observation model

$$C = c^\intercal c$$

that minimizes the trace of the error covariance matrix $K$. In addition, since we do not know the exact value for $C$, we want to iterate our $C$ such that, given an initial condition, the pair $(A, C)$ is stabilizable. The dynamics of the pseudo-observation model under the gradient flow will obey the following equation:

$$\dot{C} = \big[C, [C, M]\big]$$

where $M = KRK$ *and* $K, R$ *obey the equations*

$$A^\intercal K + KA + Q - KCK = 0$$

$$(A - CK)R + R(A - CK)^\intercal + L = 0$$

In the above equation, $Q$ and $L$ are both positive definite. In our case, we set $Q$ as our covariance matrix for the white noise and $L$ the identity matrix. The bracket stands for the commutator of two matrices and is of the form $[A, B] = AB - BA$. The intuition behind this algorithm is to flow the observation model $c$ in direction $\dot{K}$ since $K$ minimizes the error covariance matrix. We will not discuss in depth why this works in this thesis, the proof of why this method works is carefully elaborated in Professor Belabbas's paper [4]. After the pseudo solution ($C$) for the gradient flow is found, we can transfer it back to real solution ($c$) via spectral decomposition.

**Algorithm 1: Optimal sensor location**

1. *Initialize: Set N as the dimension for your device, set M the number of sensors you need. Decide a norm or SNR γ for you sensors. Decide a Hurwitz or stable matrix A for your dynamic system.*

*Decide a white noise covariance matrix Q for your system (Q needs to be positive definite). Decide a random starting point $c_0$ for the sensor location. Transform this little $c_0$ into the pseudo-observation model big C. (We start calling the real solution as the little c and the pseudo-observation model as the big C.) Specify a time T at which the algorithm might converge. In our case, we set T to be 50 seconds.*

2. *Ode45 Kth iteration:*
   - *2.1 Transform the big $C$ into a vector $V_c$ via the symmat2vec function. (The symmat2vec function converts a symmetric matrix "A" to a vector "V" of its significant part.)*
   - *2.2 Pass in the parameters A, Q, $V_c$ to the gradient function.*
   - *2.3 Transform the vector $V_c$ back to big C via the vec2symmat function. (The vec2symmat function convert the vector "V" to an N-by-N matrix "A".)*
   - *2.4 Find the solution K for the algebraic Riccati solution given all the parameters, and explicitly solve K for the equation:*

$$0 = A^{\mathrm{T}}K + KA + Q - KCK$$

   - *2.5 Find the solution R for the Lyapunovo equation given the above parameters, and explicitly solve R for the equation:*

$$0 = (A - CK)R + R(A - CK)^{\mathrm{T}} + I$$

   - *2.6 Set $M = KRK$.*
   - *2.7 Set $\frac{dC}{dt} = [C, [C, M]]$, where the bracket stands for commuting between two matrixes, explicitly $[A, B] = AB - BA$.*
   - *2.8 Integrate $\frac{dc}{dt}$ to get new observation model $C^{Kth}$ given initial point $C^{Kth-1}$ (initial point constantly updated).*
   - *2.9 Update the new initial point $C^{initial} = C^{Kth}$.*
   - *2.10 Transform $C^{Kth}$ back to $V_C^{Kth}$ by calling the symmat2vec function*
   - *2.11 Reiterate until the stopping criterion is met.*
3. *Terminate: Set $C^{K+1} = C^K$ when the stopping criterion is met. In this case we expect the solution would converge at time T.*
4. *Find the largest eigenvector $c_{opt}$ that corresponds to the largest eigenvalue in matrix C.*

This algorithm can find the optimal observation model $c_{opt}$ given a fixed norm. However, we have no information on when it will converge. Often it will converge sooner than the expected time, so it would

be wise to set an error bound on the solution. Bearing this purpose in mind, we now make some modification to the previous algorithm to have it check for error bound every once a while.

**Algorithm 2: Optimal sensor location with error bound**

1. *Specify a time step $\delta$; it should be much smaller than T. Replace T with $\delta$. Also specify an error bound $\beta$.*
2. *Initialize everything in step 1 in Algorithm 1.*
3. *Perform Algorithm 1 to solve for $V_c^{Kth}$. Take the second norm of the difference between $V_c^{Kth}$ and $V_c^{inital}$ and check if this norm satisfies the convergence condition.*
4. *Termination: If $\left\|V_c^{Kth} - Vc^{initial}\right\| < \beta$ holds, then the approximation is legitimate. If not, loop back to step 1 and set $Vc^{initial} = V_c^{Kth}$.*
5. *Transform $V_c$ back to C via the vec2symmat function.*
6. *Find the largest eigenvector $c_{opt}$ that corresponds to the largest eigenvalue in matrix C.*

This way we do not have to worry about when the solver will converge, we just need to specify a reasonable time step and error bound for the solver. In our case, we set the error bound $\beta$ to be $10^{-4}$, which is considerably small and sufficient for our experiment.

## 3.3 Numerical Result

In this section, we present the results from the two algorithms described above. Bear in mind that this is just a result for one simulation and does not generalize for all cases. In this experiment we set the dimension size to be 6, number of sensor occupied to be 3, SNR or the norm of the observation model to be 0.8. For the expected time $T$ defined in Algorithm 1, we set it to be 10. For the time step $\delta$ defined in Algorithm 2, we set it to be 2. The result of the simulation will be shown in a plot where the vertical axis represents the sensor value and the horizontal axis represents the location of the sensors.

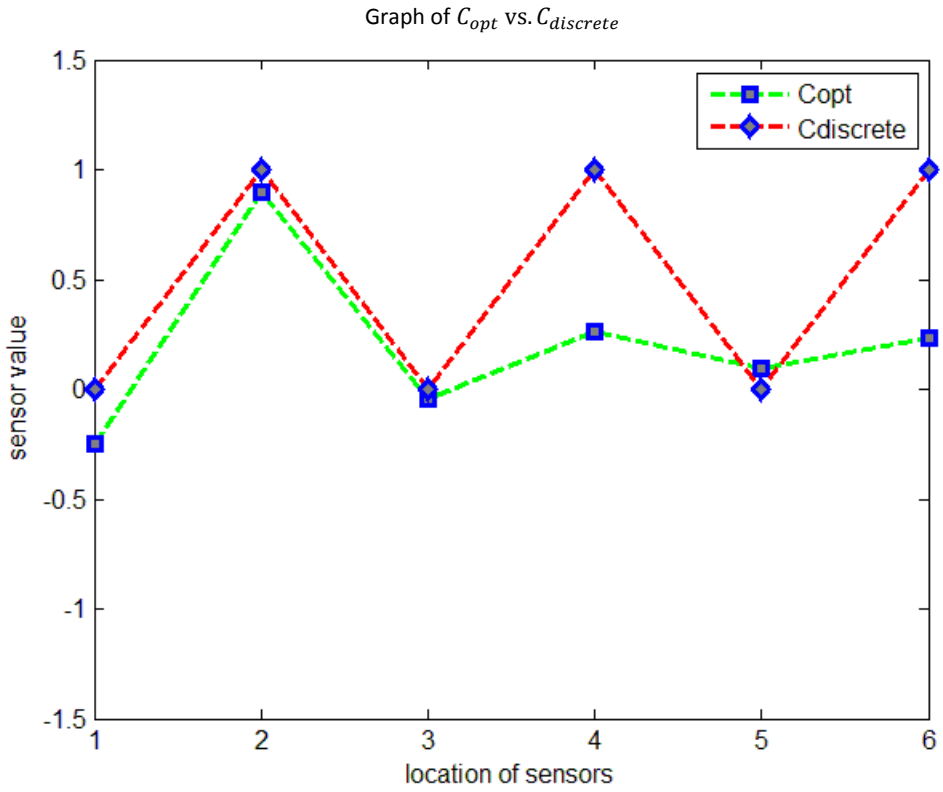


**Figure 2. Comparison between optimal observation model and optimal discrete model with algorithm 1. Dimension size 6, number of sensor deployed 3.**

In Figure 2, $c_{discrete}$ either takes 1 or 0; this is because at each location, a sensor is either present or not present. We denote 1 if the sensor is present and 0 if the sensor is not. $c_{discrete}$ is computed via comparing every trace of error covariance matrix for every combination of sensors in space. Namely, if the dimension of the space is *n* and number of sensor is *m,* then there are $\langle {n \atop m} \rangle$ number of trials we need to compute in order to find the most optimized one. Also notice that $c_{opt}$ does not consist of ones and zeros. However, if you take the 2-norm of $c_{opt}$ you will find it is 1.

For most of the time, both algorithms give out approximately the same result. Whether or not the curve for $c_{opt}$ and $c_{discrete}$ look alike really depends on the dynamics of the system. Sometimes the optimized solution for the discrete observation model can be drastically different from the solution found via gradient flow. Take a look at the graph in Figure 3 with another random dynamic system.

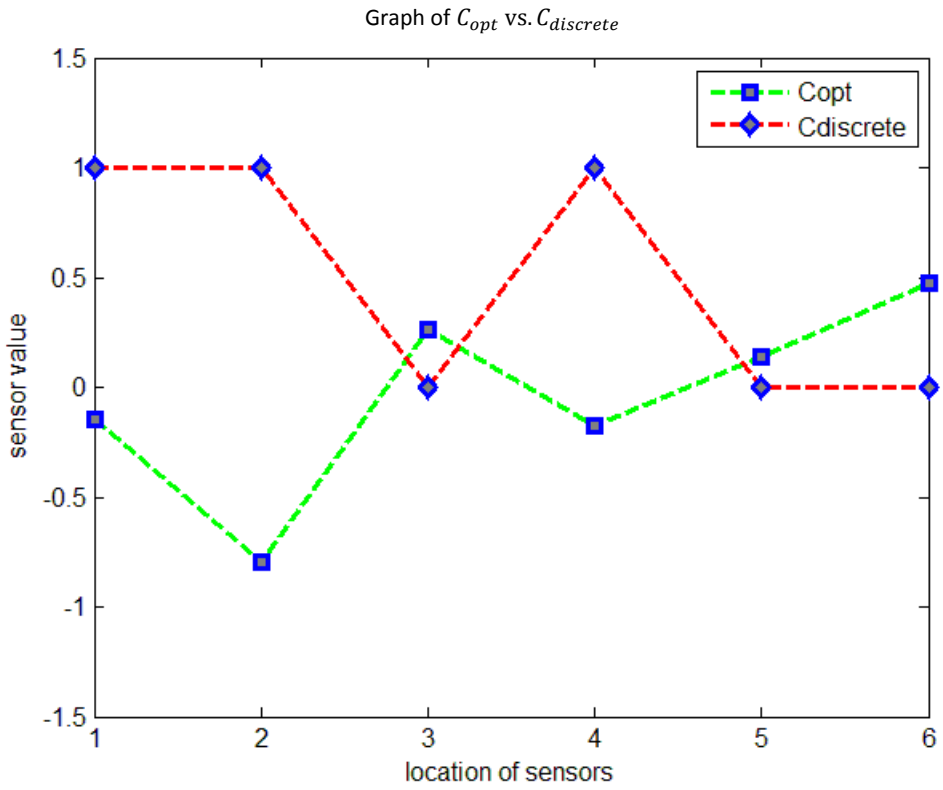


**Figure 3. Comparison between optimal observation model and optimal discrete model with algorithm 2. Dimension size 6, number of sensor deployed 3.**

Compared to Figure 2, the curve of $c_{opt}$ in Figure 3 seems to be a reversed version of the curve of $c_{discrete}$. The reason they are different is because $c_{opt}$ and $c_{discrete}$ are of a different domain. Particularly, $c_{opt}$ optimize the trace of the error covariance given a fixed norm $\gamma$ while $c_{discrete}$ optimize the trace of the error covariance given the elements in $c_{discrete}$ need to be discrete and sum up to the number of sensors. The question we want to consider next is how to construct a pseudo-optimized discrete model from $c_{opt}$ that would perform close enough or even the same as the optimized discrete model. We will discuss approaches of finding the relationship between $c_{opt}$ and $c_{discrete}$ in Chapter 5 and present the result in Chapter 6. Next we will discuss another method of finding $c_{opt}$ in Chapter 4.

# 4. Hessian Approach

## 4.1 Background Information

In this section we describe an approach for finding the suboptimal sensor location given almost the same parameters in Chapter 3. The Hessian approach evaluates the eigenvalues of the Hessian matrix coming from the trace of the error covariance matrix. If the eigenvalues of the Hessian matrix evaluated at a critical point has mixed signs, then it is a saddle point in the geometric space; if all are positive then it is a local maximum; if all are negative then it is a local minimum [20]. Since the trace of error covariance matrix came from the ARE equation it is actually complicated to solve. Instead of finding the close-form solution for the critical point, we decide to find out the solution numerically via *fsolve* in MATLAB. This means we would also need an initial sampling vector for the observation model $c$.

The second part of this algorithm is to produce the actual Hessian matrix for the trace. One way to do this is to come up with second partial derivatives for all dimensions. Unfortunately the close-form solutions for the partial derivative cannot be solved explicitly. Therefore we need to approximate these partial derivatives numerically. Details of how this is done are explained in the description of Algorithm 3.

After finding the Hessian matrix at certain critical point, we can finally evaluate the eigenvalues. This process is straightforward since we just need to go through every one of them. If one of the eigenvalues is not positive, than we need to start over again.

## 4.2 Algorithm Description

**Algorithm 3: Suboptimal sensor placement**

1. *Initialize: Set N as dimension size, M as number sensor in use. Set $\gamma$ as your signal-to-noise ratio. Setup a stable dynamic system A. Setup a disturbance matrix Q, make sure Q is positive definite. Setup a small value for $\beta$ as tolerance for your fsolve function in MATLAB because it may take a long time for the function to solve, this will speed up things without sacrificing too much accuracy. Set up an extremely small, close-to-zero negative value for $\sigma$ as the error bound for your eigenvalues because as dimensions grows, the condition that requires all eigenvalues to be positive can be difficult to achieve.*
2. *Acquire the critical point using fsolve iterations:*
   *2.1 Set up a random observation model c, normalize this observation model, and transform the little c into big C.*

*2.2 Evaluate the gradient of the trace by solving the ARE equation and Lyapunovo equation. In this case, we are evaluating* $\frac{dC}{dt} = [C, [C, M]]$*, where M is same as before and the bracket stands for commuting between two matrices.*

*2.3 Numerically solve the critical point via fsolve based on the random observation model given. In this case, since solving [C,[C,M]] is equivalent to solving [C,M], we just need to define [C,M] as a function F, then solve for this function. After solving for the solution, transform big C back to little c.*

3. *Acquire Hessian matrix at the critical point:*

*3.1 Define all possible pairs of directions of which C could move om1 and om2.*

*3.2 Compute the Hessian in the direction of one pair (om1, om2):*

*3.2.1 Find* $\delta K$ *in the direction of om1 by solving the Lyapunovo equation:*

$$0 = (A - CK)\delta K + \delta K(A - CK)^{\mathsf{T}} - K[C, om1]K$$

*where K is the solution of the ARE.*

*3.2.2 Find* $\delta R$ *in the direction of om1 by solving the Lyapunovo equation:*

$$0 = lyap\left((A - CK), \left((-[C, om1]KR - C\delta KR) + (-[C, om1]KR - C\delta KR)^{\mathsf{T}}\right)\right)$$

$$where\ x = \ lyap\ solves\ the\ Lyapunovo\ equation\ AX + XA^{\mathsf{T}} + Q = 0$$

*3.2.3 Find* $\delta Q\ by\ solving\ the\ equation$:

$$\delta Q = \delta KRK + K\delta RK + KR\delta K$$

*3.2.4 Find the Hessian in the direction of om1 and om2 by solving the following equation:*

$$\frac{\partial H}{\partial(om1)\partial(om2)} = trace\left(\left[om2, [C, om1]\right]M - [C, om2]\delta Q - trace2\right)$$

$$trace2 = 0.5 * trace([C, M] * [om1, om2])$$

*3.3 Repeat until all second partial derivatives for the Hessian matrix are found.*

4. *Evaluate the eigenvalues of the Hessian matrix at the given critical point.*
5. *Compare the eigenvalues with the error bound* $\sigma$*, if any of the eigenvalues exceed the error bound then restart from step 2.*
6. *Termination: end the loop if every eigenvalue is larger than the error bound* $\sigma$.

Algorithm 3 finds the local minimum observation model *c* for the sensor locations via picking random initial starting points. While this algorithm works, it is not without its flaws. The problem is there are no easy ways to compute the Hessian matrix evaluated at certain critical points. The current method for computing the Hessian can be costly when the dimension of the space increases since our

approximation of the second partial derivative for $J$ is still based on gradient approach. Moreover, this algorithm does not have control of how initial condition $c$ is picked every time it loops back from the beginning. One possible way to alleviate this problem or speed up the algorithm is to choose a bigger step size in the *fsolve* function and set up a limiting maximum of iterations. Another approach we suggest is to parallel compute the partial derivatives by utilizing every core we have on the modern computer. This approach can be achieved by calling the MATLABPOOL toolbox under MATLAB. However, the issue with parallel computing is that one can argue that the most native way of selecting sensors (one by one) can also be speeded up with parallel processing. Overall the Hessian approach is slower than the native method for selecting optimal sensor locations.

## 4.3 Numerical Result

In this section we present the result from the algorithm described above. We set the dimension N to be 6 and the number of sensors to be 3. For the tolerance and error bound we set it to be 0.01 and -0.01. For the SNR ratio we set it to be 0.8 which is the same as before. We also turned off the warnings in MATLAB to speed up things a bit more.

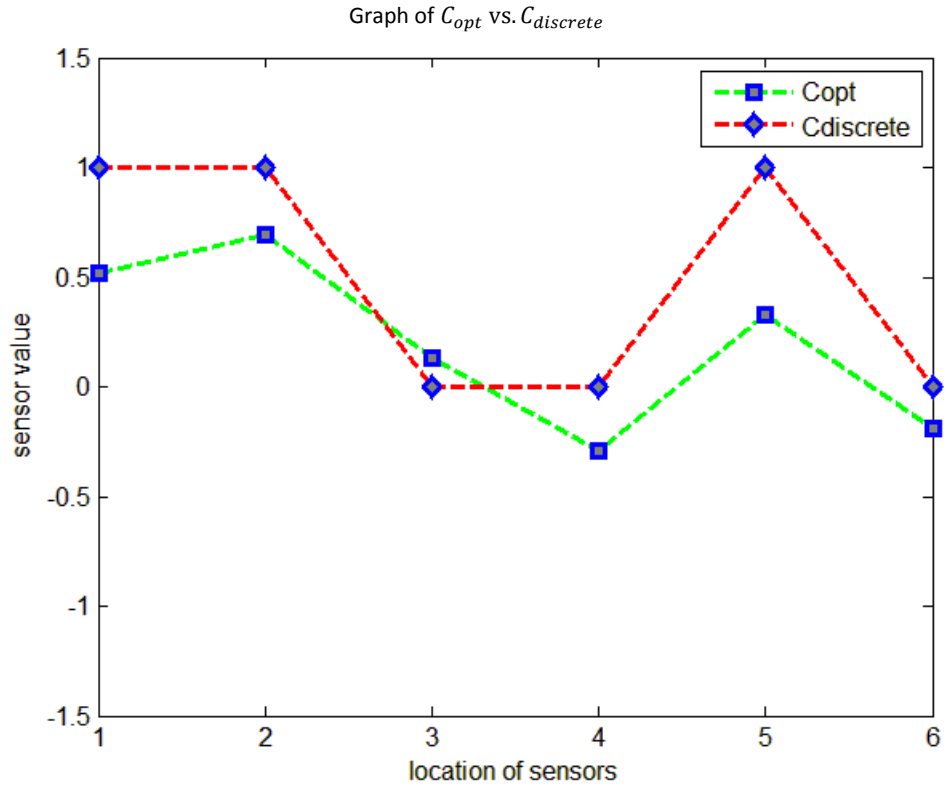


**Figure 4. Comparison of suboptimal observation model and the optimal discrete model with algorithm 3. Dimension size 6, number of sensor deployed 3.**

As one can tell from Figure 4, the suboptimal solution for the observation model closely resembles the actual discrete type of the observation model. However, there are times that this algorithm gives out inverted curves for the suboptimal observation model just like it did in Chapter 3 using the gradient approach.

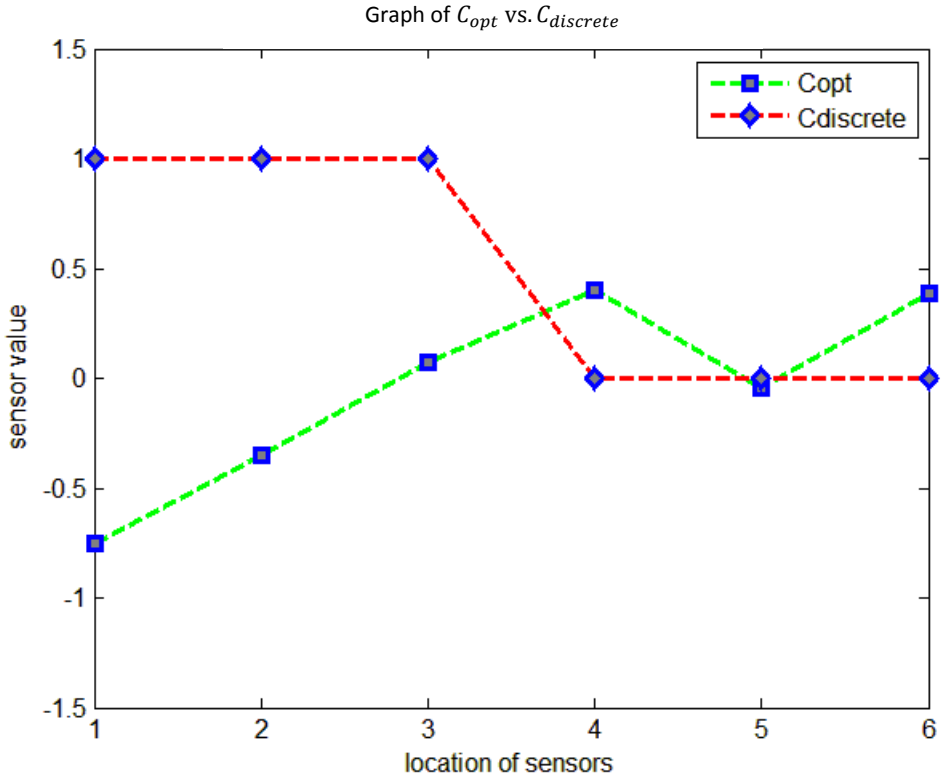


**Figure 5. Comparison of suboptimal observation model and the optimal discrete model with a different system.**

As one can tell from Figure 4 and Figure 5, how the suboptimal solution for the sensor behaves largely depends on the system. Similar to what we discussed in Chapter 3, the domain of the suboptimal solution for the sensor and the domain of discrete optimal solution for sensors are different as well. We still need to find a way to relate the suboptimal sensor solution to an optimal discrete sensor solution, especially when the plots look drastically different from each other.

# 5. Relationship between Optimal Model and Optimal Discrete Model

## 5.1 Background Information

In this section we explore different methods of finding the relationship between optimal observation model and optimal discrete model. The goal here is to find a pseudo-optimal discrete model $c_{pseudo}$ that performs as well as the optimal discrete model. In order to implement such an idea requires defining a measurement of the observer error. In one paper, Potami [21] used the $H_2$ norm of the transfer function as the measurement. The advantage of using $H_2$ norm of transfer function is that it takes consideration of the complete frequency domain rather than for a specific component to optimize the system response. The performance measurement here, however, is just the trace of the error covariance matrix since we only care about the steady-state performance of the model. As we mentioned in Chapter 3 and Chapter 4, the trace of the optimal observation model can sometimes behave drastically differently from the optimal discrete model. The result largely depends on how the dynamic system was initially set up.

## 5.2 Proposed Approaches

In this section we propose some approaches of finding pseudo-optimal discrete model that came up during the testing and simulating phase of this project. We will introduce the most native of selecting the optimal discrete observation model and then compare the solutions coming from other approaches.

**Native approach of selecting optimal discrete sensor location:**

1. *Find out all combinations given m sensors and a fixed dimension size n. There are* $\left\langle {n \atop m} \right\rangle$ *number of combinations.*
2. *For each combination* $c_{discrete(i)}$*, calculate the trace of error covariance matrix* $trace_i$*.*
3. *Find the smallest trace of error covariance through sorting.*

**Approach 1:**

The first approach we suggest is to take the first *m* largest value in the optimal observation vector then find its location by looking at the value's index. Then we flat out rest of the value in the vector to zero and replace values corresponding to those indexes we just found with ones. The new vector is thus the pseudo-optimal discrete model $c_{pseudo}$. This approach could be problematic when the plot of $c_{opt}$ looks drastically different from $c_{discrete}$.

**Approach 2:**

The second approach we suggest is to compare the optimal observation with each discrete observation model. The way to do this is to compute the Euclidian distance between them and find the discrete observation model that gives the smallest distance. Essentially we are computing this distance:

$$dis = \sqrt{\left(c_{opt} - c_{discrete(i)}\right)^2}$$

Select $c_{discrete(i)}$ such that the Euclidian distance is minimized. The selected $c_{discrete(i)}$ is thus the pseudo-optimal discrete model $c_{pseudo}$. This approach could also be problematic when the plot of $c_{opt}$ looks drastically different from $c_{discrete}$.

**Approach 3:**

The third approach is a modified version of Approach 1. The steps are described below:

1. *Take the absolute value of $c_{opt}$ and store it in vector $c_{abs}$.*
2. *Sort $c_{abs}$ from small to large and store the sorted index in vector $I_{sorted}$.*
3. *Take the last $m$ element in vector $I_{sorted}$ and find the corresponding value in $c_{opt}$. For example, if $m = 3$, then just take value $c_{opt}(end), c_{opt}(end-1), c_{opt}(end-2)$.*
4. *Sum up these value and store it in another variable $c_{temp}$.*
5. *Decide whether to flip the plot of $c_{opt}$ based on $c_{temp}$. If $c_{temp} > 0$ then leave $c_{opt}$ as it is; otherwise change the signs of $c_{opt}$.*
6. *Sort $c_{opt}$ just like in Approach 1 and find $c_{pseudo}$.*

**Conclusion:**

We eventually selected Approach 3 as our method for finding the pseudo-optimal discrete model. The main reason we selected this approach is because it deals with extreme cases when $c_{opt}$ and $c_{discrete}$ do not look alike. In addition, Approach 3 does not require prior knowledge of how $c_{discrete}$ is distributed. We will test how well it performs in Chapter 6.

# 6. Simulations and Result

In this chapter, we demonstrate the performance of our proposed approach of finding the pseudo optimal discrete model from optimal sensor model found via gradient approach. First, we discuss our experimental setup for each individual task. Then we present the accuracy of our experiment by showing the average trace of error comparison between the pseudo-optimal discrete model and the optimal discrete model. Finally we look at the runtime of the gradient flow method compared to the simple iterative way of selecting the optimal discrete model. We will not be comparing the Hessian approach in this section because it is still gradient flow based, so surely it will not run any faster than the gradient flow approach.

## 6.1 Experimental Setup

We will be conducting two experiments, one is with the number of sensor fixed and the other one is with the space dimension fixed. For the first experiment, we will be conducting a total number of ten thousand experiments on each dimension size and take the average of the trace of error covariance matrix. Each experiment initializes a random starting observation model $c_{initial}$ and a random stable matrix A, therefore each experiment is different. There are two traces of error covariance matrixes we are looking at here; one trace corresponds to the pseudo-optimal discrete model and the other one corresponds to the optimal discrete model. To measure the performance of our model, we take the average ratio of the two traces across all dimensions. If the model performs well then the curve should be flat and close to 1. For the second experiment, which is very similar to the first experiment, we will be conducting the same number of experiments except with the space dimension fixed.

To accelerate our procedure of simulating, we utilize the parallel computing capability in MATLAB by calling the MATLABPOOL function. The procedure is done by running multiple functions that returns both traces simultaneously but with a different dimension/sensors size.

## 6.2 Trace of Error Comparison (Number of Sensor Fixed)

In this section we compare the trace of error covariance for three different cases. Let us take a look at how the trace ratio behaves for one of the test case. In this example we set the dimension size to be 6 and the sensor deployed to be 3.

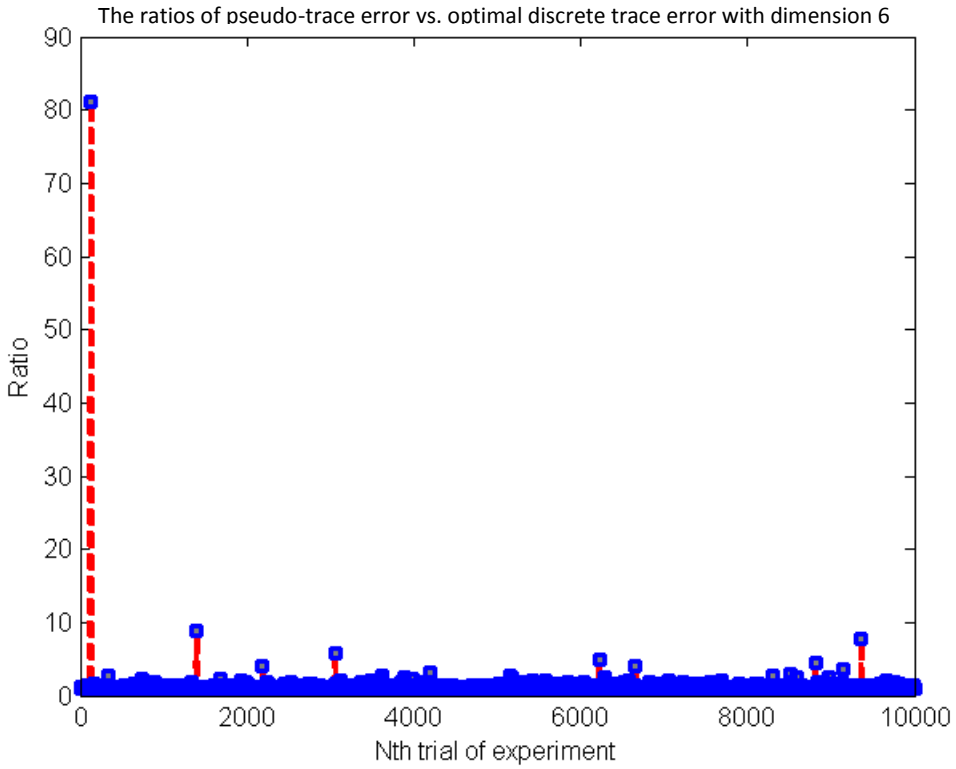


**Figure 6. Graph showing the ratio between the two traces: trace1/trace2, where trace1 is the trace obtained by the pseudo-optimal discrete observation model and trace 2 is the trace obtained by the optimal discrete model.**

As one can tell from the Figure 6 here, the ratio between two traces is actually very close to 1. This indicates the performance between the pseudo-optimal discrete observation model and the optimal discrete model is very close. There are occasional cases where the ratio between the two traces bumps up to very high. But such cases are rare so we can ignore them for most of the time. To conclude our result, we take a look at the Table 1.

**Table 1 Comparison of algorithm performance for fixed sensor size 3**

| Dimension | Average Ratio | Average Cov($c_{pseudo}$) | Average Cov($c_{discrete}$) |
|---|---|---|---|
| 6 | 1.0498 | 5.8694 | 5.4893 |
| 7 | 1.0582 | 7.4114 | 6.6700 |
| 8 | 1.0579 | 8.9723 | 8.0527 |

Table 1 shows that the performance of the algorithm is consistent across all three dimensions. The trace of the error covariance also grows with the dimension size, which matches our prediction as well.

## 6.3 Trace of Error Comparison (Dimension Fixed)

In this section we take a look at how the trace of error compares for a different number of sensors given that the space dimension is fixed. The same setup is used as in last section. We set the dimension size to be 6 but with the sensor varying from 1 to 3.

**Table 2 Comparison of algorithm performance for fixed dimension size 6**

| Number of sensors | Average Ratio | Average Cov($c_{pseudo}$) | Average Cov($c_{discrete}$) |
|---|---|---|---|
| 1 | 1.0253 | 6.0055 | 5.8258 |
| 2 | 1.1438 | 7.0983 | 5.3829 |
| 3 | 1.0553 | 5.8580 | 5.4797 |

Table 2 shows that the performance of the algorithm is consistent, even with a different number of sensors selected.

## 6.4 Runtime Comparison

In this section we compare the runtime of gradient flow with the native way of selecting optimal discrete observation model. We basic follow the setup in Section 6.2, but we only compare the runtime for each case.

**Table 3 Runtime comparison**

| Dimension | Runtime by gradient flow | Runtime by native selection |
|---|---|---|
| 6 | 0.0749 | 0.0146 |
| 7 | 0.0895 | 0.0340 |
| 8 | 0.1002 | 0.0675 |
| 10 | 0.1268 | 0.1973 |
| 11 | 0.1348 | 0.3366 |

As we can tell from Table 3 that the runtime for solving the solution grows as the dimension space grows. Another piece of information we may extract from the runtime comparison table is that before dimension 10, it is wise to use the native method for selecting the optimal discrete sensor location. However for any dimension bigger than 10, it is best use the gradient flow approach to solve for the optimal discrete sensor location since it takes less time and the performance of such model would not be so different than the native method.

# 7. Conclusions and Future Work

In this thesis, we presented a fully automatic system for finding the optimal sensor location and translating it to discrete type model that can be used in real world. Given a fixed SNR parameter and a stabilized system, our algorithm will: (i) find the optimal sensor location in continuous time, (ii) find the pseudo-optimal discrete sensor location, and (iii) compare the difference between them. Our result demonstrates we are able to find a pseudo-optimal discrete model which performs similarly to the optimal discrete model. There are only a few times when extreme cases happen that the trace of the error covariance generated from the pseudo-optimal discrete model drifted far away compared to the error covariance generated by the optimal discrete model.

As for runtime, the gradient approach, if not considered any modification, is much slower than the native way of finding the optimal discrete model. However since we are adopting Algorithm 2, which is a modified version of the gradient approach with many conditions relaxed, we are actually able to see Algorithm 2 outperforms the native way of finding the optimal discrete model when the dimension of space grows to a certain point. It seems that when the dimension of space is small, it is favorable to use the native way of picking the optimal discrete model. When dimension space grows, it is however favorable to use the modified gradient approach to find the optimal observation model.

There are some changes we could make to improve the accuracy of our design. One change we could make is to provide a more complex white noise error covariance matrix. Instead of assuming the white noise error covariance matrix to be stationary and only associated to the space dimension size, we can make it time variant and have it associated to more parameters, such as the number of sensors or sensor types.

Finally, we could limit our initial observation model to models that correspond to discrete time space in order to accelerate the process of finding the optimal sensor placement. After all, it is the optimal discrete model that we care about. However, limiting the initial condition to a certain domain may not lead to improved efficacy in some cases because the optimal sensor location in those cases may all be negative. It may be more fruitful to find a way to distinguish which kind of system produces an abnormal observation model and which kind of system that does not.

# References


[1] R. Kalman and R. Bucy, "New results in linear filtering and prediction theory," *Journal of Basic Engineering*, vol. 83, no. 1, p. 95, 1961.

[2] G. Wredenhagen and P. Bélanger, "Curvature properties of the algebraic Riccati equation," *Systems & Control Letters*, vol. 21, no. 4, pp. 285-287, 1993.

[3] T. Basar, S. Meyn and W. Perkins, *Lecture Notes on Control System Theory and Design*. 2014, p. 181.

[4] M. Belabbas, "Geometric methods for optimal sensor design," unpublished.

[5] F. Fahroo and M. Demetriou, "Optimal actuator/sensor location for active noise regulator and tracking control problems," *Journal of Computational and Applied Mathematics*, vol. 114, no. 1, pp. 137-158, 2000.

[6] H. T. Banks and F. Fakhroo, "Legendre-tau approximations for LQR feedback control of acoustic pressure fields," *J. Math. Systems Estimation Control,* vol. 5, no. 2, pp. 271-274, 1995.

[7] H. T. Banks, S. L. Keeling, R. J. Silcox, "Optimal control techniques for active noise suppression," in *Proceedings of the 27th IEEE Conference on Decision and Control,* 1988, pp. 2006-2011.

[8] Fariba Fahroo, "Optimal placement of controls for a one-dimensional active noise control problem," *Kybernetika*, vol. 34, no. 6, pp. 655-665, 1998.

[9] K. Morris, "Linear-quadratic optimal actuator location," *IEEE Transactions on Automatic Control*, vol. 56, no. 1, pp. 113-124, 2011.

[10] E. Musulin, C. Benqlilou, M. Bagajewicz, L. Puigjaner, "Instrumentation design based on optimal Kalman filtering," *Journal of Process Control*, vol. 15, no. 6, pp. 629-638, 2005.

[11] G. Heyen, M. Dumont, B. Kalitventzeff, "Computer-aided design of redundant sensor networks," in *Proceedings of the 12th European Symposium on Computer Aided Process Engineering,* 2002, pp. 685-690.

[12] J. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, 1975.

[13] D. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, MA, 1989.

[14] E. Musulin, M. J. Bagajewicz, J. M. Nougue´s and L. Puigjaner, "Instrumentation design and upgrade for principle components analysis monitoring," *Ind. Eng. Chem. Res.,* vol. 43, no. 9, pp. 2150–2159, 2004.

[15] K. Herring and J. Melsa, "Optimum measurements for estimation," *IEEE Transactions on Automatic Control*, vol. 19, no. 3, pp. 264-266, 1974.

[16] R. Bansal and T. Başar, “Simultaneous design of measurement and control strategies for stochastic systems with feedback,” *Automatica*, vol. 25, no. 5, pp. 679-694, 1989.

[17] Y. Liu, J. Slotine and A. Barabasi, “Observability of complex systems,” *Proceedings of the National Academy of Sciences*, vol. 110, no. 7, pp. 2460-2465, 2013.

[18] R. L. Stratonovich, “Application of the theory of Markov processes for optimum filtration of signals,” *Radio Eng. Electron. Phys. (USSR),* 1:1-19, 1960.

[19] R. Murray, *Optimization-Based Control*, 1st ed. 2010, pp. 2-3.

[20] J. Rosenberg. (2009). *Critical Points of Functions of Two and Three Variables* [Online]. Available: http://www2.math.umd.edu/~jmr/241/crits.html

[21] R. Potami, “Optimal sensor/actuator placement and switching schemes for control of flexible structures,” Ph.D dissertation, Worcester Polytechnic Institute, 2008.